**CISC 275 – Spring '12**
**Lab 2 – Swing/JUnit**

(All source files available from TA webpage)

**Non-graded portions**

Save HelloWorldGUI.java and import the file into an Eclipse project. (Feel free to import into your old HelloWorld project from Lab 1, or create a new project):

- From within Eclipse's Package Explorer view (left pane), right click the "src" folder from the desired project and select "Import..."
- General → File System, then hit "Next"
- Browse to the correct directory, then check off the files to import on the right hand side
- Click "Finish", and the source file is now part of your project

This file is an introduction to Swing, a Java GUI environment. It creates a JFrame, initializes it with a simple label "Hello World", and draws it to the screen. Study and run this code.

**Graded Exercise – On your own OR in pairs**

1. Create a new project in Eclipse, say, "Lab2". Import the 3 remaining source files into the new project (i.e. *Celsius*.java).

   Notice instead of creating a JFrame from within main(), as in the HelloWorldGUI example, we simply extend JFrame with the CelsiusConverterGUI class.

   Also notice that our "View", the GUI class, is separate from our "Controller", the main class. See comment at top of CelsiusController.java for how this represents the Model View Controller (MVC) design pattern.

   Study and run the code to test it out.

2. Add new functionality to the Celsius Converter:
   - Add a drop-down list offering at least two extra conversion options (e.g., meters to inches, kgs to lbs) in the GUI
   - Add new static conversion methods to the controller class
   - Program should make appropriate conversion based on option selected in drop-down list.

   Hints: Use the Swing Tutorial, linked from TA webpage. "Using Swing Components" is the relevant section. Especially JComboBox!
   Use the Java API to learn about the classes you are using. Get in the habit of doing this.
   Use the code that is already there as a guide. For example, add an ActionListener, and corresponding callback function, for your JComboBox in the same way as exists for the JButton... .
   Consider renaming things to be more generic: "celsiusLabel" to "inputLabel", "fahrenheitLabel" to "outputLabel", etc. The Eclipse "Refactor" menu can help with this.

3. TestCelsiusController.java contains JUnit tests for the controller methods.
   - First, let's set up your project to use JUnit. Right click your project and click Properties (or go to menu Project → Properties). Choose Java Build Path on the left, and then the Libraries tab on the right. Choose Add Library... then JUnit, Next, choose version 4 and then Finish.
   - Next, let's run the test code that is there. Right click the file, Run As → JUnit Test

- It purposely fails to show you what happens in this case.  Delete or modify the offending test, and run again to see what changes.
- Add tests for your new static methods in the controller class

4.  Via Sakai, submit the three updated files with your name(s) clearly listed after the @author tag at the top of every class.